NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

**VIRTUAL PROTOTYPING
AS AN AID FOR
CONTROL SYSTEM ANALYSIS**

by

Arnold P. Selnick
September, 1994

Thesis Advisor: Isaac I. Kaminer

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September, 1994	Master's Thesis

4. TITLE AND SUBTITLE
VIRTUAL PROTOTYPING AS AN AID FOR CONTROL SYSTEMS ANALYSIS

5. FUNDING NUMBERS

6. AUTHOR(S)
Arnold P. Selnick

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT *(Maximum 200 words)*

Virtual Prototyping provides an opportunity for control engineers to observe and evaluate their designs in a "real world" setting without the costs or risks associated with flight test. *Designers Workbench* is a computer aided design tool that allows a user to create, view, modify and animate three-dimensional databases. These may include instruments, avionics displays and any *out-the-window* applications such as runways or terrain. The dynamic behavior of specific elements can be observed by *linking* data to the desired element. In this way a realtime animation of the simulation can be generated. The animation provides another analysis tool for the designer, by representing data in a more intuitive environment.

Data files from two separate controller simulations was used. Cockpit instrumentation was modelled and used for each animation to monitor the aircraft states. In addition, each animation had an *out-the-window* perspective to view the aircraft model as it flew the prescribed trajectory.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Virtual Prototyping, Designers Workbench, Control System Design, Real-time Animation	49
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

Approved for public release; distribution is unlimited

Virtual Prototyping
as an Aid for
Control System Analysis

by

Arnold P. Selnick
Lieutenant, United States Navy
B.S. Penn State University, 1987

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

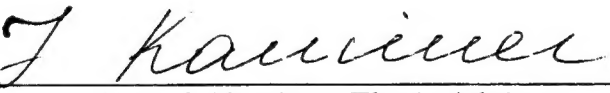
from the

NAVAL POSTGRADUATE SCHOOL

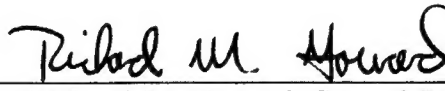
September , 1994

Author: 

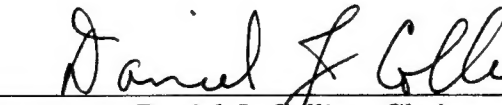
Arnold P. Selnick

Approved by: 

Isaac I. Kaminer, Thesis Advisor



Richard M. Howard, Second Reader



Daniel J. Collins, Chairman
Department of Aeronautics and Astronautics

ABSTRACT

Virtual Prototyping provides an opportunity for control engineers to observe and evaluate their designs in a "real world" setting without the costs or risks associated with flight test. *Designers Workbench* is a computer aided design tool that allows a user to create, view, modify and animate three-dimensional databases. These may include instruments, avionics displays and any *out-the-window* applications such as runways or terrain. The dynamic behavior of specific elements can be observed by *linking* data to the desired element. In this way a realtime animation of the simulation can be generated. The animation provides another analysis tool for the designer, by representing data in a more intuitive environment.

Data files from two separate controller simulations was used. Cockpit instrumentation was modelled and used for each animation to monitor the aircraft states. In addition, each animation had an *out-the-window* perspective to view the aircraft model as it flew the prescribed trajectory.

TABLE OF CONTENTS

I. INTRODUCTION 1

A. BACKGROUND 1

B. DESIGNER'S WORKBENCH 1

C. CONTROLLER SIMULATIONS 2

1. Auto-Land Controller for the F-14 3

2. Trajectory Tracking Controller for a UAV 4

II. MODELLING BASICS 6

A. THE GRID 6

B. STRUCTURE 6

C. REGIONS 8

1. Perspective Region 8

2. Clip Region 9

III. FILE FORMAT 12

A. .DATA FILES 12

B. .VARS FILES 13

C. .DRT FILES 15

IV. LINKING 16

A. CREATION OF A LINK 16

B. TRANSLATION LINKS 18

C. ROTATION LINKS 20

D. EYE POINTS 21

1. Tail-following Eyepoint 22

2. Interactive Eyepoints 24
V. ANIMATIONS 28
A. LINK EDITOR 28
B. REALTIME MODULE 28
1. Performance 29
VI. CONCLUSIONS 31
APPENDIX A. MODELS CREATED WITH DWB 32
INITIAL DISTRIBUTION LIST 39

LIST OF FIGURES

1.1 Auto-land Controller: Trajectory and Altitude Error 3

1.2 Trajectory Controller Flight Path 4

1.3 Trajectory Controller Errors 5

2.1 Structure Page for Attitude Indicator 7

2.2 Perspective Region Structure and Attribute Page 10

2.3 Clip Region Structure and Attribute Page 11

3.1 Modified .data file 13

3.2 Corresponding .vars file 14

4.1 Nested Link 1 16

4.2 Nested Link 2 17

4.3 Nested Link 3 17

4.4 Coordinate Link 18

4.5 Link for Altitude Error 19

4.6 DWB Rotation 20

4.7 Rotation Link for Attitude Indicator 21

4.8 DWB Eyepoint Perspective 22

4.9 Tail-following Geometry 23

4.10 Interactive View Choice Buttons 24

4.11 Momentary Link for Rear View Button 26

4.12 Tail-follow Eyepoint with Discrete Definitions 26

4.13 Discrete Value Mappings for Angle Change 27

4.14 Discrete Mappings for Altitude Controller 27

A.1 Designer's Workbench Link Editor Environment 32

A.2 USS Midway, CV-41 33

A.3 Generic Jet Aircraft 34

A.4 Aircraft and Carrier Initial Positions 35

A.5 Generic Cockpit 36

A.6 Trajectory Controller : Aircraft Initial Position on Airfield 37

ACKNOWLEDGMENT

I would like to take this opportunity to thank Dr. Isaac Kaminer for his encouragement in pursuing this project and for his support and extreme patience. I would also like to thank Glen Raudins at Coryphaeus Software, without his vast technical support, none of this would have been possible.

I. INTRODUCTION

A. BACKGROUND

While SIMULINK and SYSTEMBUILD provide powerful capabilities in the design, testing and analysis of dynamic, non-linear controllers, they are very limited in their abilities to provide suitable data representation for conducting an actual flight. The UAV project currently at the Naval Postgraduate School (NPS) plans to provide Over-the-Horizon capabilities to a battlefield commander. Virtual prototyping will be able to provide a link between a pilot on the ground and the UAV out of visual range. This link will provide real-time feedback of the aircraft states in a more intuitive setting than two- or three-dimensional plots.

The primary goal of this thesis is to expand on previous work done at the NPS Avionics Lab that incorporated a new software package with the current simulation and design tools already being used. With the software, a user is able to create models in a "virtual" environment and generate animations of these models.

For future simulations, a model of a generic cockpit was created, and the process of generating an animation was streamlined to provide future users with a relatively quick and easy handbook.

B. DESIGNER'S WORKBENCH

Designer's Workbench (DWB) is a computer aided design tool that allows users to create, view, modify and animate three-Dimensional graphic databases. These may include any combination of instruments or avionics displays for *in-the-cockpit* views, and any *out-the-window* applications such as runways and terrain.

Simulation data from a controller can be *linked* to specific elements within DWB to define the dynamic behavior of that element. In this way an animation can be generated. This visual representation can be used as a more intuitive way to evaluate the performance of the flight control system.

C. CONTROLLER SIMULATIONS

Data from two different controller simulations was used for demonstrating the capabilities of DWB. The first was an Auto-land controller for an F-14 Tomcat designed by LCDR Rob Niewoehner (Ref. 6). Glideslope and angle-of-attack were controlled with power, stabilators and dynamic DLC. The animation required the creation of two models for the *out-the-window* view: an aircraft carrier and an aircraft to represent the Tomcat. Cockpit instrumentation was also developed to monitor the states of the aircraft. The states were simulation specific, based on the needs of the designer, and DWB was used to create the appropriate instrumentation. In this case, airspeed, altitude, vertical velocity, attitude and power were monitored. A generic instrument to visually display errors in commanded altitude versus actual altitude, was also created. In addition, links were included to allow the user to interactively choose from a number of *eye-point* perspectives, allowing for different views during the animation.

The second data file used was from a trajectory controller simulation for the NPS UAV, Bluebird, designed by LT. Eric Hallberg (Ref. 4). While a DWB animation had already been created by LCDR Mark Lagier, (Ref. 5) the file was modified to include the generic instrument models for the *in-the-cockpit* view and included a tail-following eyepoint perspective along with the interactive eye-point links. Errors in position and altitude were also displayed.

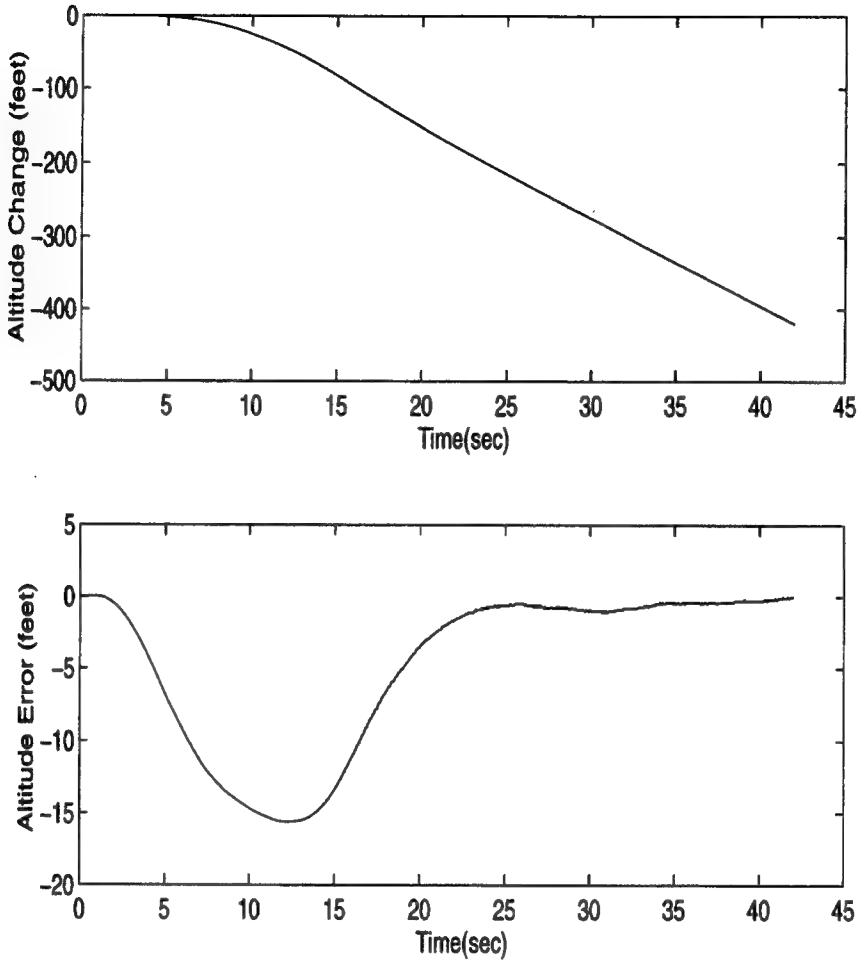


Figure 1.1: Auto-land Controller: Trajectory and Altitude Error

1. Auto-Land Controller for the F-14

For the simulation, the aircraft was initialized at 1200 feet AGL and was commanded to descend at 600 feet/min, as is done in an actual approach. No disturbances were introduced. The data file was cut from 90 seconds of simulation time to 40 seconds to improve DWB performance, and as Figure 1.1 shows, the DWB animation only goes through a 400 foot altitude change.

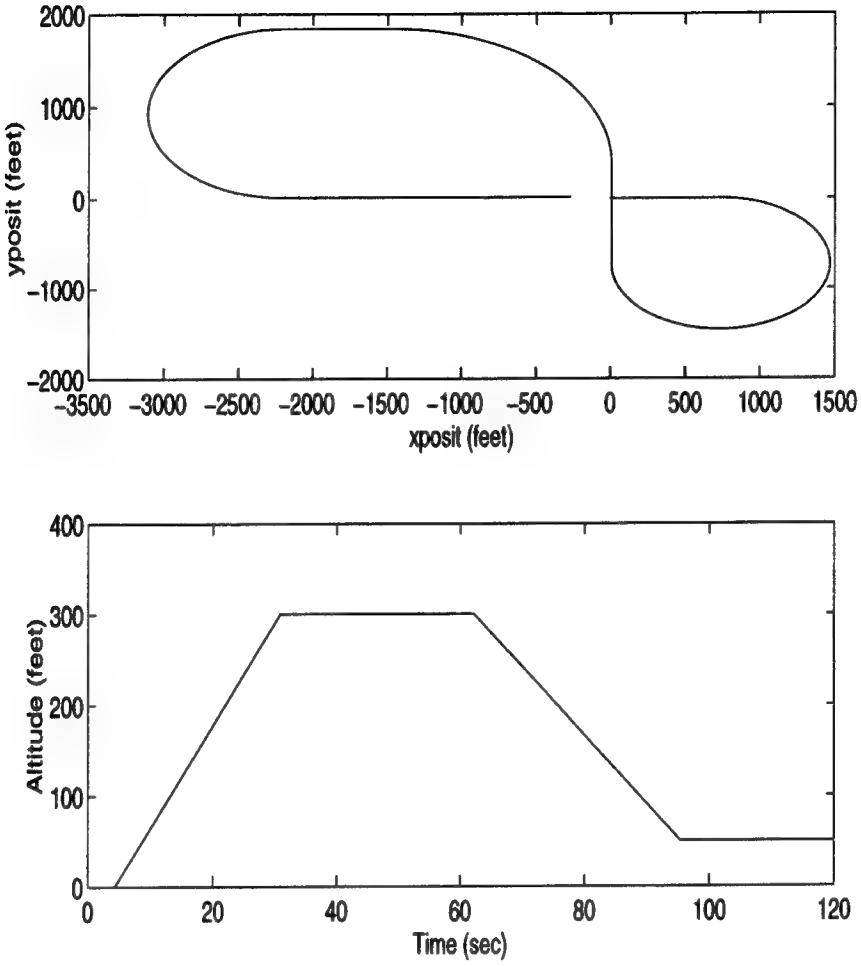


Figure 1.2: Trajectory Controller Flight Path

2. Trajectory Tracking Controller for a UAV

Figure 1.2 shows the figure-8 flight path that the UAV was commanded to fly, climbing to an altitude of 300 feet, and returning to its start point at 50 feet above the ground. A right crosswind was introduced at the start as a disturbance that the controller had to account for while trying to track the commanded flight path. The *disturbance* was removed on the short final approach to the field, again to evaluate the performance of the controller. Figure 1.3 shows the errors between the commanded and actual trajectory.

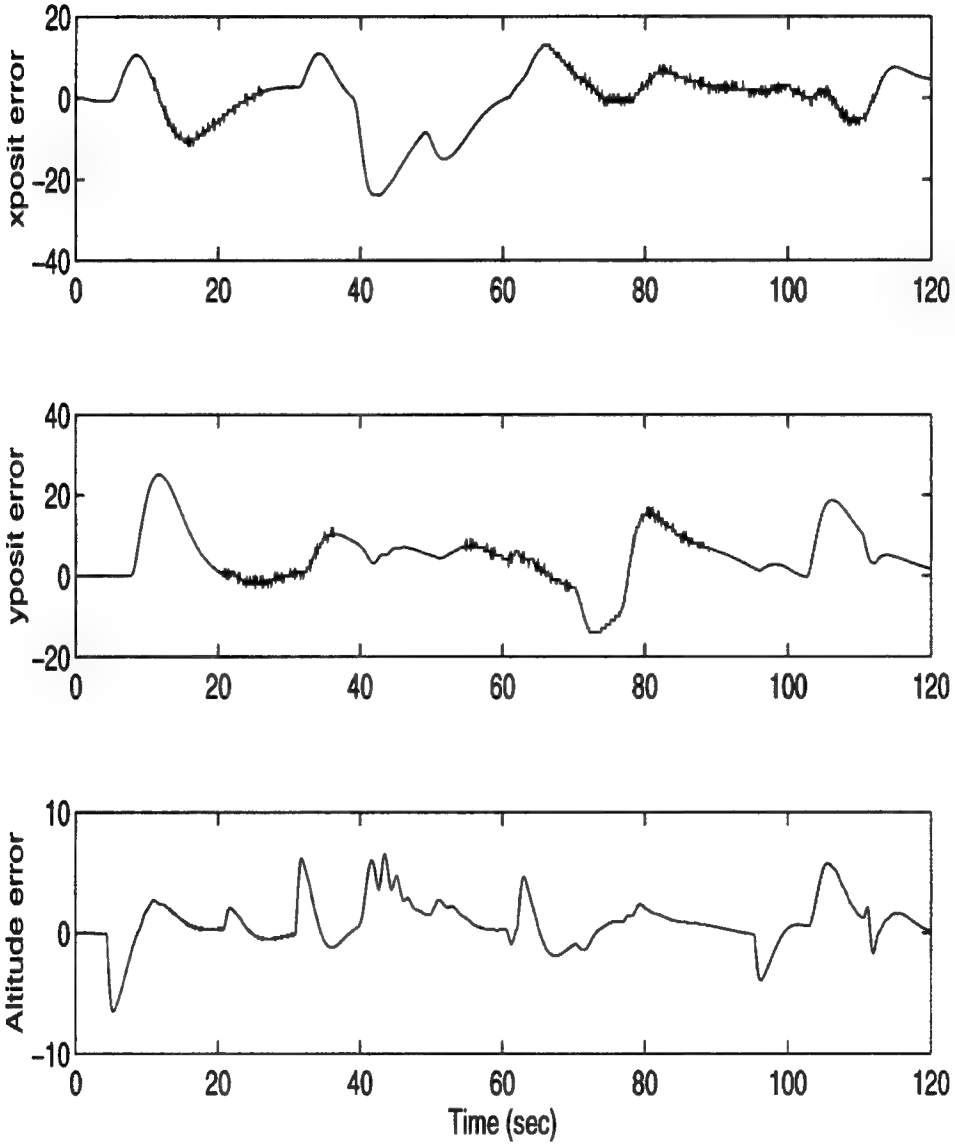


Figure 1.3: Trajectory Controller Errors

II. MODELLING BASICS

A. THE GRID

The three-dimensional database that is created in DWB is initially drawn on a 2-dimensional grid. This grid can be positioned or oriented anywhere in space using the mouse. DWB defaults to a grid with an XY orientation, with a standard size and units (either metric or English). These values can be changed, but whatever the choice, the user needs to be consistent with the grid spacing since these values become important during the linking process.

For *out-the-window* models, it is recommended that the user change the orientation of the grid to *XZ*, since this also helps the linking process. For *in-the-cockpit* models, the grid default should be used. For more discussion on the grid orientation, see Chapter IV.

Most of the *editing icons* used for constructing a model are self explanatory. The modelling process itself is not complex, but it is very time consuming. The models constructed for all the animations are shown in the Appendix. Finally, it is very important that the user be aware of the grid orientation during any editing operation (such as scaling, translating or creating a polygon), or the desired effect will not be achieved.

B. STRUCTURE

DWB provides an option called Z-buffering which gives the viewer a realistic 3-dimensional view of an animation. There is a penalty in processing time and not all models, such as a static, cockpit need it. Here, the structure of the model becomes

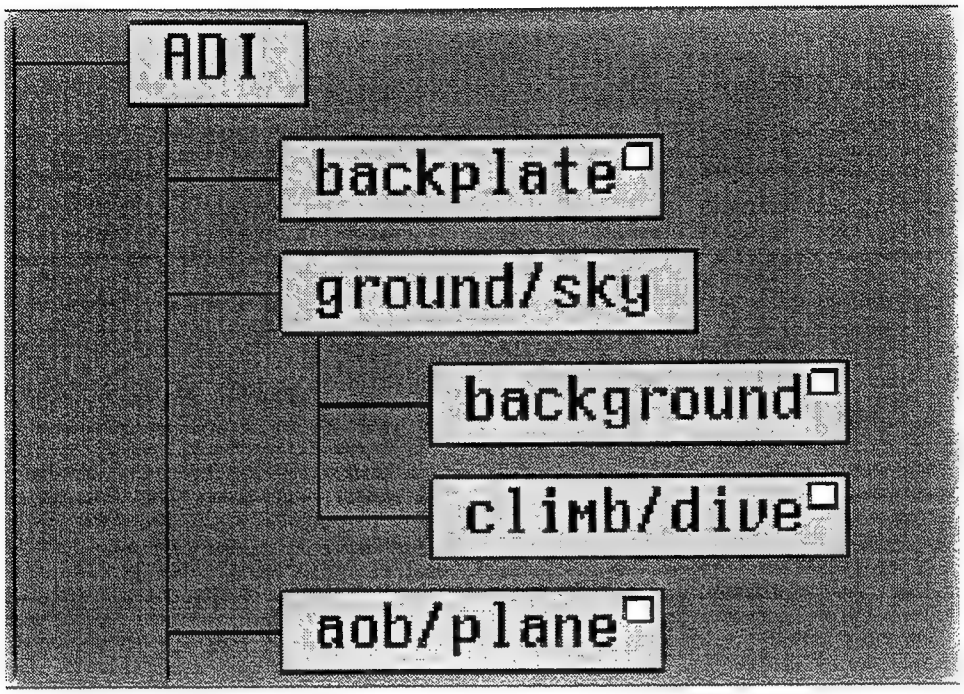


Figure 2.1: Structure Page for Attitude Indicator

important. DWB automatically generates hierarchic databases composed primarily of groups which are made up of polygons.

Grouping your model is important for a number of reasons:

- Ease of editing
- Linking in Perspective or Clip regions
- Drawing Routines

Editing is much easier if objects are logically grouped and the groups and subgroups have meaningful names. This becomes very important in large databases with numerous polygons.

Logically grouping your database will also help when creating links after the modelling is complete. This is especially important in *Clip* or *Perspective* Regions, where it is important to know which group contains the elements in these defined

areas. It is also easier to select an element in the structure page by its name. The drawing routines are important for the appearance of the model. DWB automatically draws the polygons and strings from top to bottom of the hierarchy created. This could cause an element to be blocked by another drawn above it, so care must be taken to put the groups in the correct order. Figure 2.1 shows the structure page for the Attitude indicator model. Here, DWB first drew the backplate, then the ground/sky background on top, and so on down the the list. A correct view can easily be obtained by editing with the *Structure* icon and placing the groups in the order needed. Again, structure editing is important for objects that do not need Z-buffering to preserve the correct perspective.

It should be noted that the *background* and *climb/dive* groups are included within the *ground/sky* group, which is defined as a *Clip Region* , described in the next section.

C. REGIONS

1. Perspective Region

A perspective region provides for an *out-the-window* view of an animation. Once the boundaries of a perspective region have been designated, any other DWB file can be imported. Inside the perspective region, the objects and eye-points can move however the user desires, while outside, the perspective remains static. It is important that the user does not delete the separate file used for the perspective region once it is imported. The *parent* file containing the perspective region does not actually save the file within the region, but imports it whenever initially opened. This means that changes that need to be made in the perspective region, must be made within the original file.

To generate a Perspective Region:

- With the filename designated as *Parent*, create a new group (Click on *NEW*), and then designate it as the *Parent.*

  * Note: The *Parent* flag tells the software which group you want to make changes or add groups to.

- Within the *Edit* icon, select *Modify Attributes*

- Click on *Normal* and select *Perspective Region* then *Select Reference Points*

- Follow the prompts to determine boundaries for the region and make sure to apply the changes

- Under the *Structure* icon , select *Create* and *Ext. Page*

- Select the file that will be imported into the region

It should be noted that once the file is imported, it will probably not be shown with the correct perspective. Select *Modify Attributes* again, and modify the X,Y,Z rotations and eyepoint positions. The eyepoint positions must be at (0,0,0) for linking. The background color, Field of view and aspect ratio were probably also changed and can also be modified on this page . Any other modifications must be made to the separate, **external file itself** , not while it is included as an import file.

2. Clip Region

It may be desired that a model contain an element that need only be shown within a specified area, and covered up outside of that boundary. The airspeed and altitude tapes were two such elements where the user need only view a partial segment

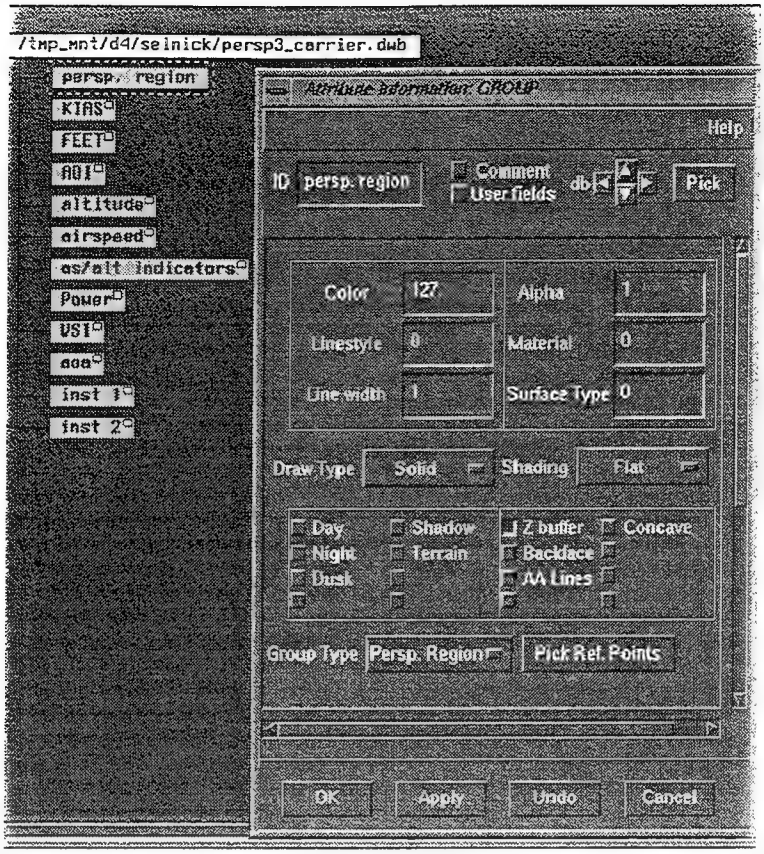


Figure 2.2: Perspective Region Structure and Attribute Page

of the element. Once the tapes were developed, they were placed in another group called a *Clip Region.*

To create a Clip region:

- The elements need to be contained in a separate group.
- The group that contains the element, must be designated the *Parent.*
- Within the *Edit* icon, select *Modify Attributes*
- Click on *Normal* and select *Clip Region* then *Select Reference Points*

DWB will then give a prompt to choose the lower left and upper right

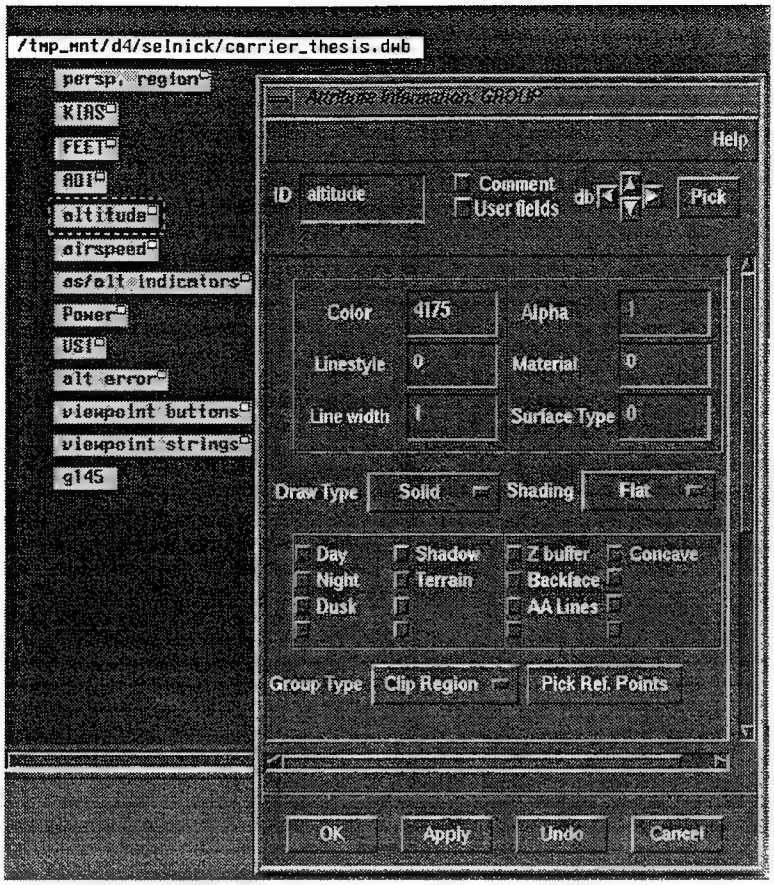


Figure 2.3: Clip Region Structure and Attribute Page

boundaries of the clip region. The region will now effectively block any portion of the element outside the defined boundary, and if the element translates out of the defined region, it will disappear.

Note that *Altitude* is selected in Figure 2.3 because this is the group where the *Clip region* is defined and it contains within it, a subgroup where the actual altitude tape model is located.

III. FILE FORMAT

To run an animation in DWB, two external files are needed: a *.data* file and a *.vars* file. The data file drives the display and the vars or variable definition file correlates the variable in the data file with those in the linking process.

A. .DATA FILES

The data file that drives a simulation may be in both binary and ASCII format. In both controllers tested, an ASCII file generated from a MATLAB simulation was used. The file must contain 51 columns of data, but can be of any length. The first column must contain the simulation step number and it must be an integer.

Neither of the data files used was originally in this format, and had to be modified. Step numbers were added in sequence, in column 1 to every row of data and zeros were added to increase the file to 51 columns. The problem of the first column not being an integer still remained. This was corrected by using a program called *mat_dwb.exe* , on the Avionics' Lab PC, America.

To Modify the data:

- Determine the length of the .data file
- On PC America: *cd* \ *Borlandc* \ *bin* then *ac100svr* to allow for file transfer.
- On workstation: *ftp america* then *put (data filename).data*
- On PC:*quit* to end file transfer, then type *mat_dwb* to start the conversion program.

Step Index	Fwd Velocity	Lateral Velocity	Vertical Velocity	Roll Rate ....
1	7.330e+01	0.000e+00	0.000e+00	0.000e+00
2	7.330e+01	4.113e-10	-7.362e-04	1.924e-09
3	7.330e+01	5.431e-09	-1.131e-03	9.670e-09
4	7.330e+01	2.266e-08	-1.400e-03	2.003e-08
5	7.329e+01	5.783e-08	-1.652e-03	2.886e-08
6	7.329e+01	1.122e-07	-1.895e-03	3.360e-08
7	7.329e+01	1.824e-07	-2.070e-03	3.330e-08 ....
8	7.328e+01	2.607e-07	-2.092e-03	2.822e-08
9	7.327e+01	3.377e-07	-1.879e-03	1.941e-08
10	7.327e+01	4.030e-07	-1.373e-03	8.311e-09
11	7.326e+01	4.476e-07	-5.440e-04	-3.505e-09
12	7.325e+01	4.651e-07	6.092e-04	-1.459e-08
13	7.325e+01	4.519e-07	2.064e-03	-2.377e-08
.				
.				
.				
.				

Figure 3.1: Modified .data file

- Follow the prompts then type *ac100svr* again
- On workstation, you may have to close then open the ftp utility again , and *get* the modified data file.

* Note: Figure 3.1 shows a portion of the data file after it has been modified. The column labels do not appear in the actual file.

B. .VARS FILES

There is a limit to the number and type of variables that can be defined in a *.vars* file if using an ASCII data file, but these should more than enough for most simulations. They are:

- Floating point (150)

```
1
float u (forward velocity ft/sec)
float v  (lateral velocity ft/sec)
float w    (vertical velocity)
float p   (roll rate rad/sec)
float q   (pitch rate rad/sec)
float r   (yaw rate rad/sec)
float phi      (*57.3 deg-rad, -90 to +90)
float theta (*57.3 deg-rad, -90 to +90)
float psi      (*57.3 deg-rad, -90 to +90)
float x_pos    (ft)
float y_pos     (ft)
float z_pos    (ft)
float elevator  (rad positive-pitch down)
float rudder    (rad  positive-left yaw)
float aileron    (rad positive-right roll)
float throttle   (unitless 1 is 4 hp)
float x_cmd      (ft)
float y_cmd      (ft)
float z_cmd      (ft)
float airspeed   (ft/sec-true)
float wind_x     (ft/sec)
float wind_y     (ft/sec)
float wind_z     (ft/sec)
float x_err
float y_err
float z_err
```

Figure 3.2: Corresponding .vars file

- Integers (50)
- Character Strings (50)

When writing a file, each line can contain only one variable definition and must include the type and variable name. The comments that are in parentheses are optional. The *.vars* file corresponding to Figure ??is shown in Figure 3.2. The position of the variable definition within the file is extremely important as it directly relates to the position of that variable in the data file.

During the linking process, DWB may not list the variables in the order they were written, since it tries to put them in alphabetical order.

The easiest way to create a new *.vars* file is to just copy an old one and change the variable names that are specific to the new data file.

C. .DRT FILES

These files are used in the *Realtime* module (RTM) option and contain a combination of geometry and link files in a binary format. Animations run in the link editor (LE), emulate those run in RTM, but are used primarily for quick and easy verification and testing of the dynamic display. RTM animations run without the editing environment, and have optimized drawing and transformation routines.

Once all the modelling is complete and the links have been defined in the link editor, the file can be saved as a *.drt* file. An RTM animation can now be run simply by selecting *Run RTM* under the *Animation* icon.

IV. LINKING

A. CREATION OF A LINK

The linking process allows the user to take a data file and map specific variables within the file to certain elements within the 3-D database, and use this mapping function to drive the animation. To create a link, simply select the desired element within your database, then select *Create/Edit* on the *Link* icon. Next, select a mapping function and variables that will define the link.

DWB allows for numerous functions to be used for link definition. Figures 4.1 through 4.3 show how more complex functions can be created by nesting the definitions. By selecting *mapped* instead of *variable* within the Link definition page,

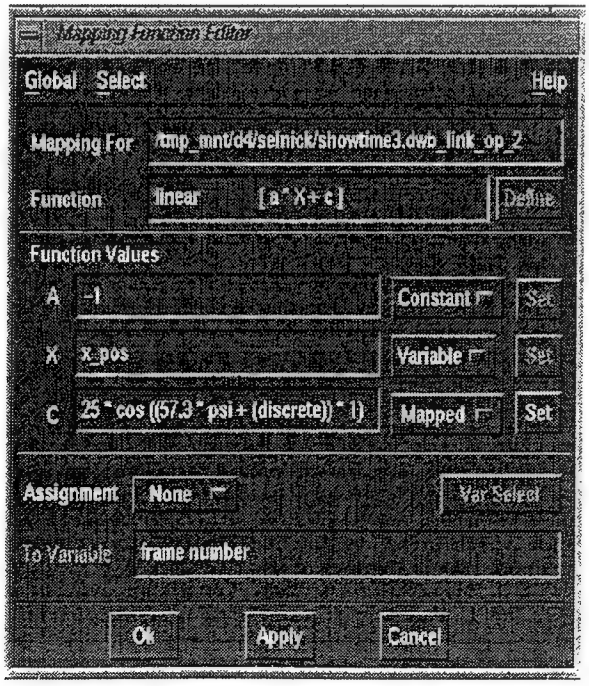


Figure 4.1: Nested Link 1

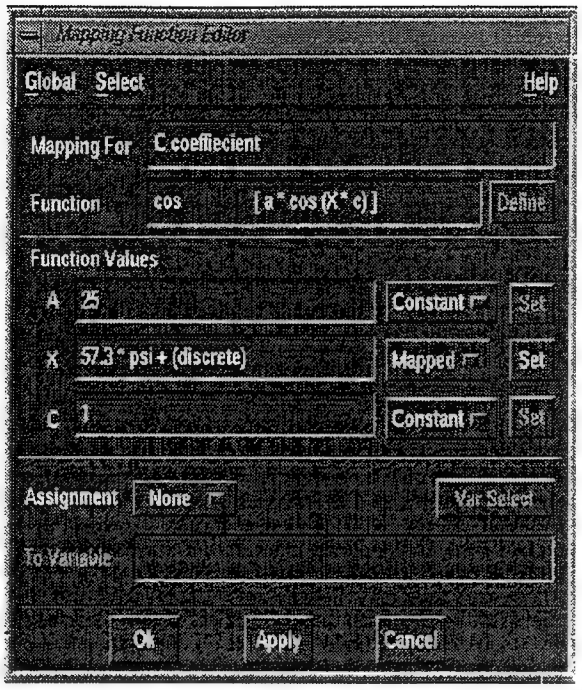


Figure 4.2: Nested Link 2

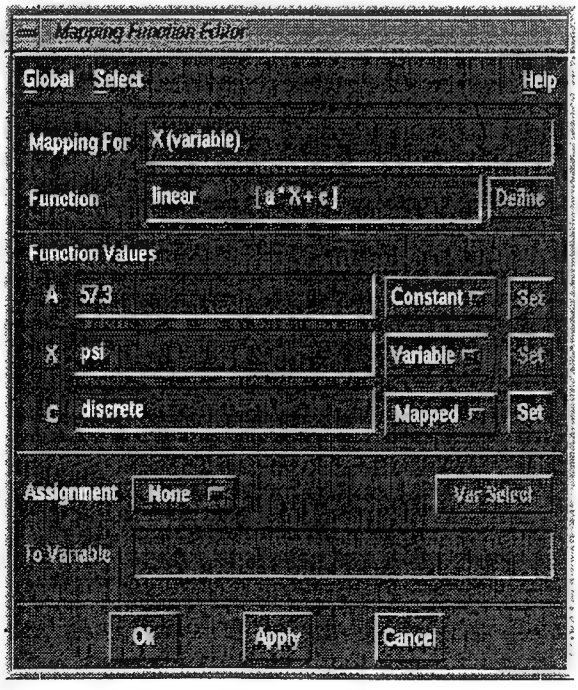


Figure 4.3: Nested Link 3

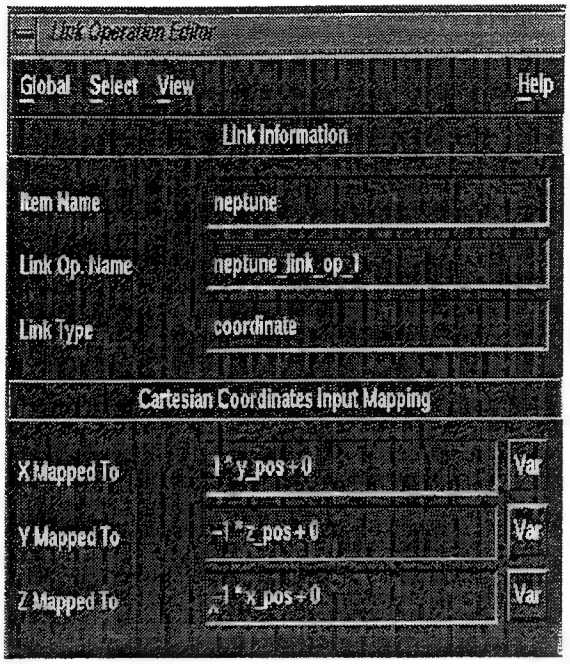


Figure 4.4: Coordinate Link

that one "variable" can now be defined as an entire new function.

B. TRANSLATION LINKS

As with all links, it is very important that the user know the orientation and size of the grid and the grid spacing. The *Translation* link is a relatively simple one and is commonly used to map external $x - y - z$ positions from a data file to the aircraft model. The variables that define the motion of the aircraft or element to be moved, must be linked to the proper axis on the defined grid. The actual value of the variable will be used to translate the linked object that same number of units across the grid. Measurement units such as 'feet' or 'cm' do not matter as long as it is to scale and all files are consistent. This is a *relative* translation, and it is based on an elements relative position on the grid.

Figure 4.4 shows the *coordinate* link used to move the UAV model within the

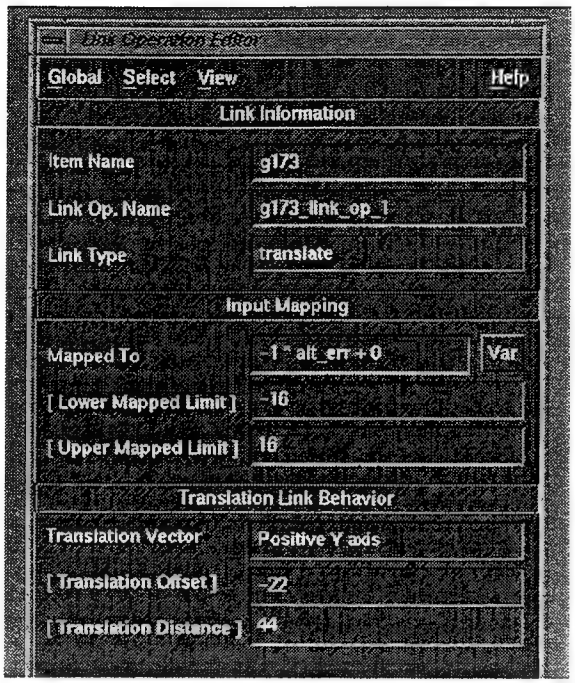


Figure 4.5: Link for Altitude Error

grid. The link is simply three translations defined at once. The variables *xpos, ypos* and *zpos* are from the MATLAB simulation data and are used to define the aircraft position on the grid. Note the grid axis the variables are *mapped* to. The reason for this mapping will be discusssed in the *Eyepoint* section.

It may also be necessary to scale a translation link based, for example, on the size of the display versus the range of the variable used. Figure 4.5 is an example of how to define limits to a translation link using the instrument modelled for displaying altitude error for the Altitude Controller simulation.

- The altitude error was determined from the MATLAB plots, to range from $\pm$ 16 feet (from Fig. 1.1)
- The instrument model was built on a grid with XY orientation, so the indicator must be translated up and down the *positive Y-axis.*

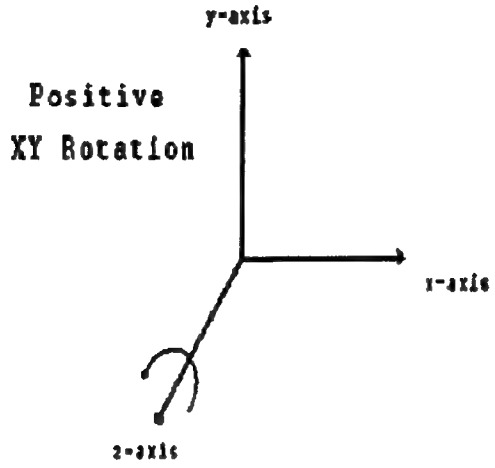


Figure 4.6: DWB Rotation

- The indicator is 44 grid units high, so the *Translation Distance* can only be 44.
- The range of motion is based on the objects initial position. A *Translation Offset* is used to initially place the indicator at the minimum value which is 22 grid units down the Y-axis.

C. ROTATION LINKS

Once again, it is a necessity for the user to know the grid orientation used for linking. DWB handles angular rotation as degrees. The *Rotation axis* is based on the *right hand rule* as shown in Figure 4.6. In Figure 4.7, roll or ϕ, is mapped based on the defined grid. Translation is in the negative Z direction and roll would about the DWB defined *positive XY axis*, or the Z-axis. Values for pitch and heading were similarly mapped.

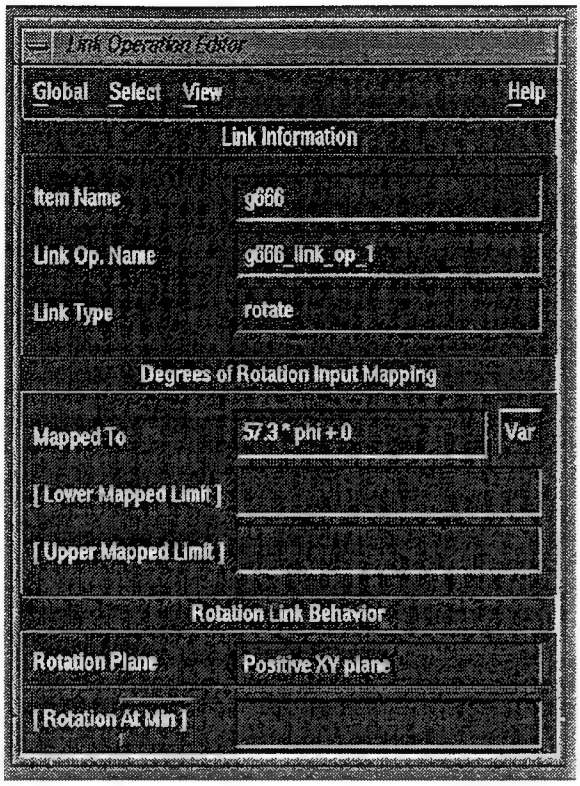


Figure 4.7: Rotation Link for Attitude Indicator

D. EYE POINTS

Eyepoint links are used to follow a specific element, like an aircraft, during an animation. DWB uses the axis system shown in Figure 4.8 for the eyepoint perspective and for eyepoint link definitions. When defining the link, if an object or eyepoint is to move left and right on the screen, the variable defining that motion must be mapped to the X-axis. An object moving into the screen must be mapped to the Z-axis and the Y-axis is used for motion up or down the screen. This is why it is easier to model *out-the-window* scenes on a grid with an *XZ* orientation. When an *XY* orientation is selected with the edit icons, DWB automatically puts the user looking directly down the negative Z axis (The default eyepoint axis). If the model was drawn with another initial orientation, the translations necessary to look down

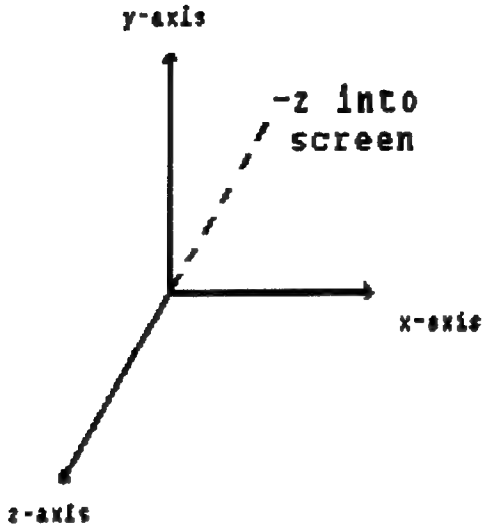


Figure 4.8: DWB Eyepoint Perspective

the Z axis would not be exact, and the eyepoint links would probably not work!

The *eyepoint* links are also a relative motion link. The initial eyepoint position defaults to the (0,0,0) coordinate of the grid when an animation is begun. The motion of the eyepoint link is based on grid size but not on the grid orientation. Again, the direction of motion is based on the axis in Figure 4.8.

When creating an eyepoint link, the filename or page in the perspective region must be selected and then *Create/Edit* selected under the *Link* icon. Eye-point position or rotation must then be selected.

1. Tail-following Eyepoint

To achieve an eyepoint that would follow an aircraft throughout the simulation, simple geometry was used. The example in Figure 4.9 shows how the links for the position controller were defined. First, an arbitrary distance for the eyepoint to remain behind the plane, D , must be chosen. The variables *xpos* and *ypos* were

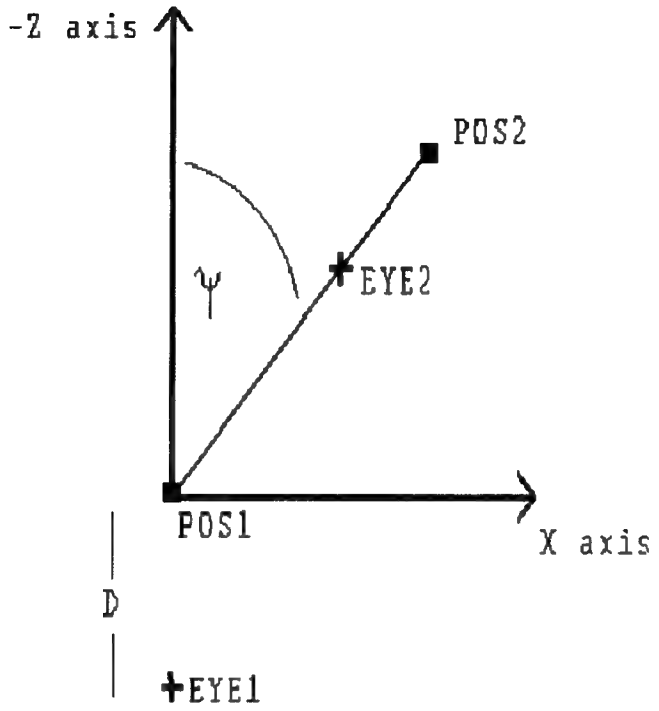


Figure 4.9: Tail-following Geometry

defined in the simulation to govern the motion of the plane in an XY-plane. Again, for an eyepoint when viewing the animation, this plane of motion becomes XZ ! As the aircraft model moves from POS1 to POS2 , the eyepoint must move from EYE1 to EYE2 to remain at the chosen distance behind the model and maintain that "tail-following" perspective. So, as the model moves *xpos* units on the negative Z-axis and *ypos* units on the positive X-axis, it is easy to see that the eyepoint position is governed by the following equations:

- $EyeZ = -xpos + (D * cos\psi)$
- $EyeX = ypos - (D * sin\psi)$

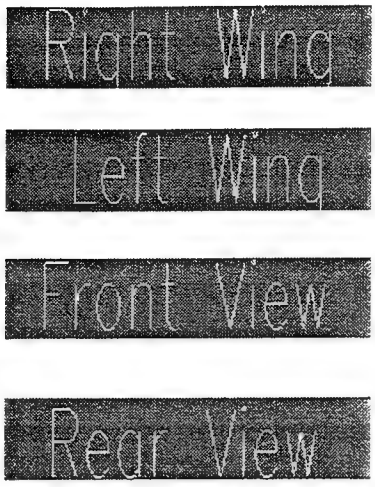


Figure 4.10: Interactive View Choice Buttons

2. Interactive Eyepoints

It may be desired to have a number of different eyepoints defined that the user can choose interactively while the animation is running. For each of the *View Choice* buttons in the position controller simulations, shown in figure 4.10, a *Momentary Control* link was defined. When the user selects a particular view with the mouse, a discrete value of either 0,1,2 or 3 is generated. The user defines these discrete values within the link.

These values are then used in the eyepoint position and rotation links where they were defined to correspond to certain angles. These angles were added to the heading rotation angle,ψ , and in this way positioned the eyepiece for the desired perspective.

The discrete definitions, corresponding angle changes and resulting views are shown below:

Discrete Value	*Angle Change*	*View*
0	0	Rear
1	90	Left Wing
2	180	Front
3	270	Right Wing

Figures 4.11 through 4.13 show the actual *Momentary Link* for the *Rear* view button and how they were integrated within the eyepoint links. When creating the momentary link, select *discrete* as the function to produce the box on the left of figure 4.11. When *Define* is selected, the right figure appears, allowing the user to input any desired values, in this case, zeros.

Figure 4.12 shows where the discrete mapping would be added to the actual eyepoint link. The values now used for these discrete mappings are shown in figure 4.13, and are added to the heading angle.

Figure 4.14 shows the slightly different mappings used for the Altitude Controller Animation.

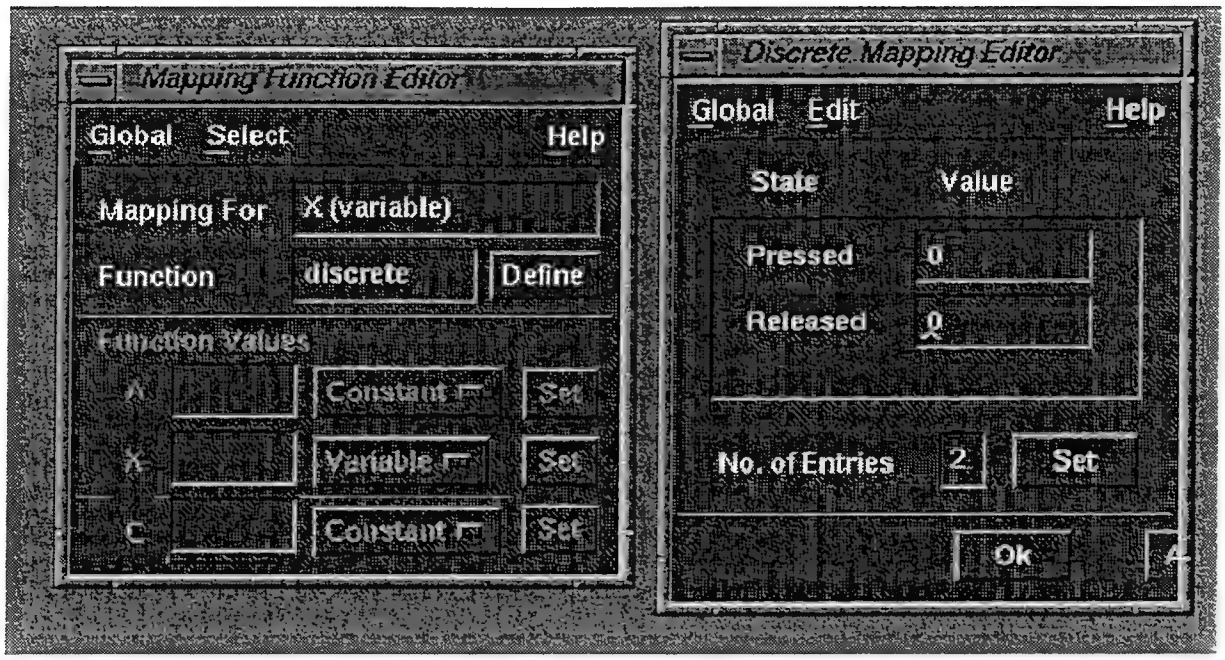


Figure 4.11: Momentary Link for Rear View Button

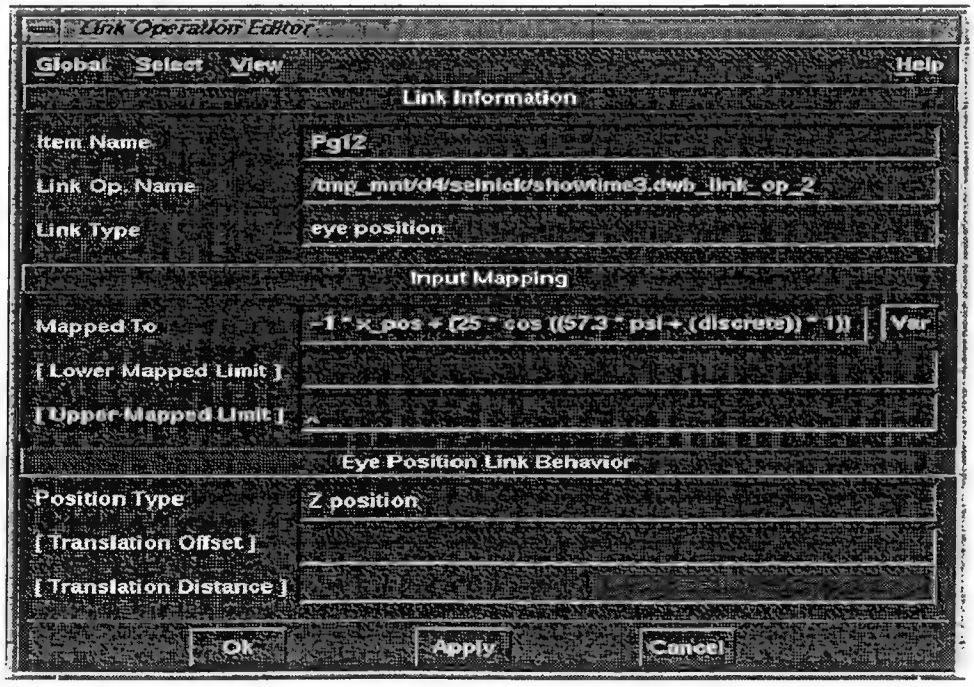


Figure 4.12: Tail-follow Eyepoint with Discrete Definitions

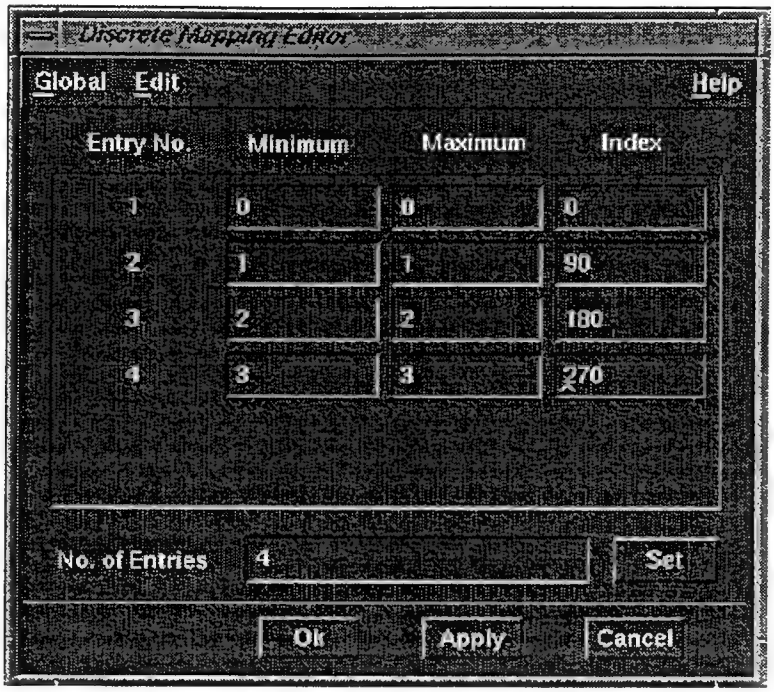


Figure 4.13: Discrete Value Mappings for Angle Change

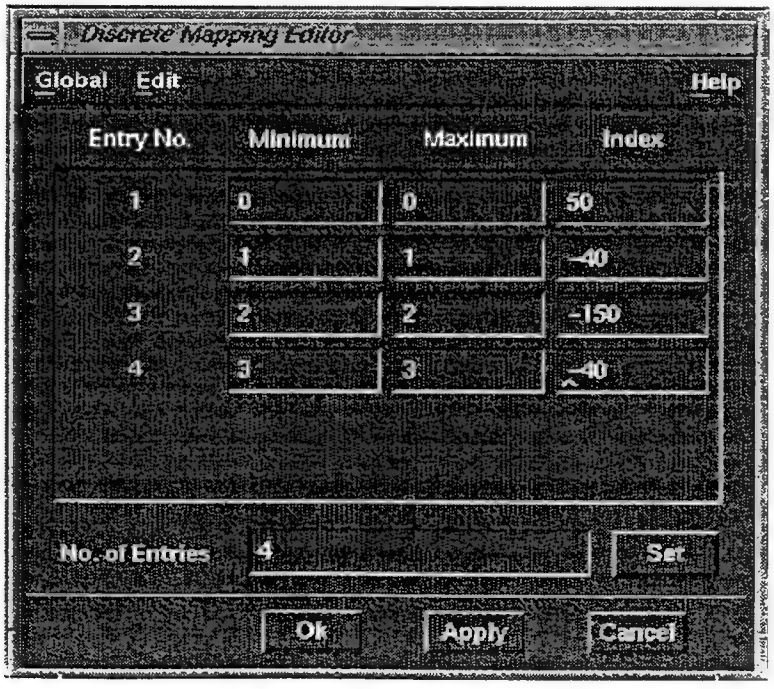


Figure 4.14: Discrete Mappings for Altitude Controller

V. ANIMATIONS

A. LINK EDITOR

While animations run in the *Link Editor* are not done in an optimum way, they do allow for quick verification and testing. To begin an animation, the following files must be loaded:

- The *.lnk* file created for dynamic behavior.
- The *.vars* file, found under *Load Sim Names* within the *Links* icon.
- The *.data* file, found under *Configure Comms* within the *Animation* icon.

  Select *Data File* under *Methods of Comm* .

  Enter desired file.

Note : Once the .data file is entered in Configure Comms window, all the necessary files can be saved as a *.cco* file. The *Link Auto Load* function can then automatically load the *.vars* , *.lnk* and *.data* files. This option is available if the user desires, whenever the modeled is subsequently loaded. **All the files must have the same name as the model file.** The animation can then be started once the loading is complete.

B. REALTIME MODULE

Once the editing for an animation is complete, an optimum animation can be generated using the *Realtime Module* (RTM). The RTM manager processes the

animation at a faster rate and optimizes certain functions, like z-buffering, to improve the realtime performance.

To generate an RTM animation,

- The file must first be saved an a *.drt* file.
- Select *Configure Display*, under the *Animation* icon, and select desired options. Most likely, Z-buffer ON or SELECTIVE.
- Select *Run RTM*.

The animation will now take up the entire screen, eliminating the Link Editor environment. The *esc* key can be used to return to the editing environment.

1. Performance

Actual realtime performance of the animation is not guaranteed with DWB and is highly hardware and software dependent. Optimal development of the database can improve simulation time by allowing the RTM drawing routines to operate more efficiently. A few examples of how to improve DWB performance are:

- Use Selective Z-buffering. DWB then only processes those elements that have the flag set on the *attributes* page. Elements such as cockpit instruments where only one view is needed would not have this flag set.

  Note : Z-buffer flashing is caused by elements that are constructed on the same plane on top of one another, such as a runway over the airfield. DWB cannot decide which perspective is correct. It alternates priorities as the perspective changes, producing a flashing effect. This can easily be avoided by translating each element slightly off the original plane , each at different distances.

- Use the minimum number of polygons to create a model. DWB must update all polygons each time the scene is updated.
- Use flat illumination. This is processed the fastest even though it may not provide the most realistic image.

The NPS Avionics Lab currently has relatively fast graphics processors. The greatest contributor to long simulation time is the length of the data file being used. Trial and error has shown that ten to fifteen lines of data per second of simulation produce real-time animations within the link editor, and slightly faster during RTM, without losing much quality in the animation itself.

VI. CONCLUSIONS

With its many capabilities, DWB provides another valuable analytical tool for Control System Designers. While unable to provide a detailed analysis of the design performance, it can provide limited, but important feedback to the designer in the more intuitive graphical environment. With the capability to read data over the Ethernet, future use for DWB will be to provide real-time feedback for "Hardware-in-the-loop" simulations for the UAV project at NPS, and eventually be used for tethered or wireless flight testing.

APPENDIX A

MODELS CREATED WITH DWB

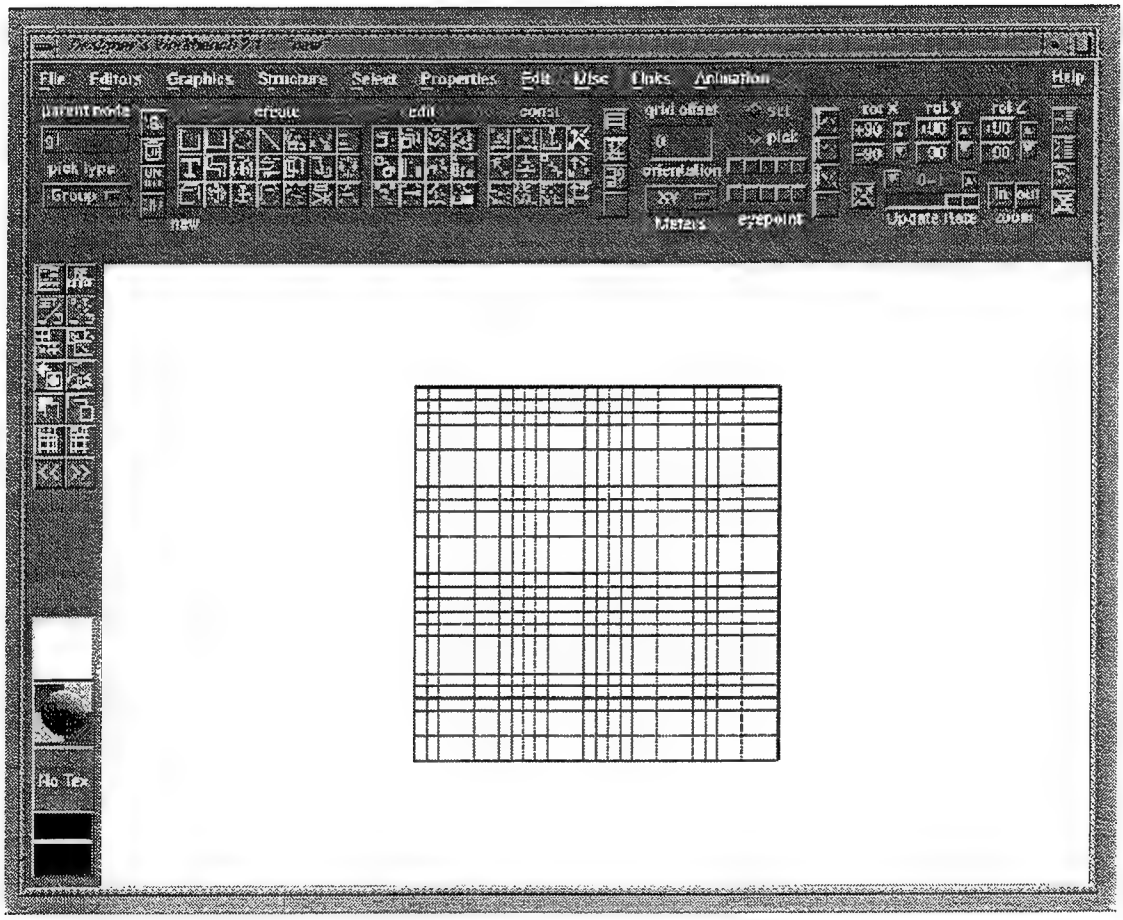


Figure A.1: Designer's Workbench Link Editor Environment

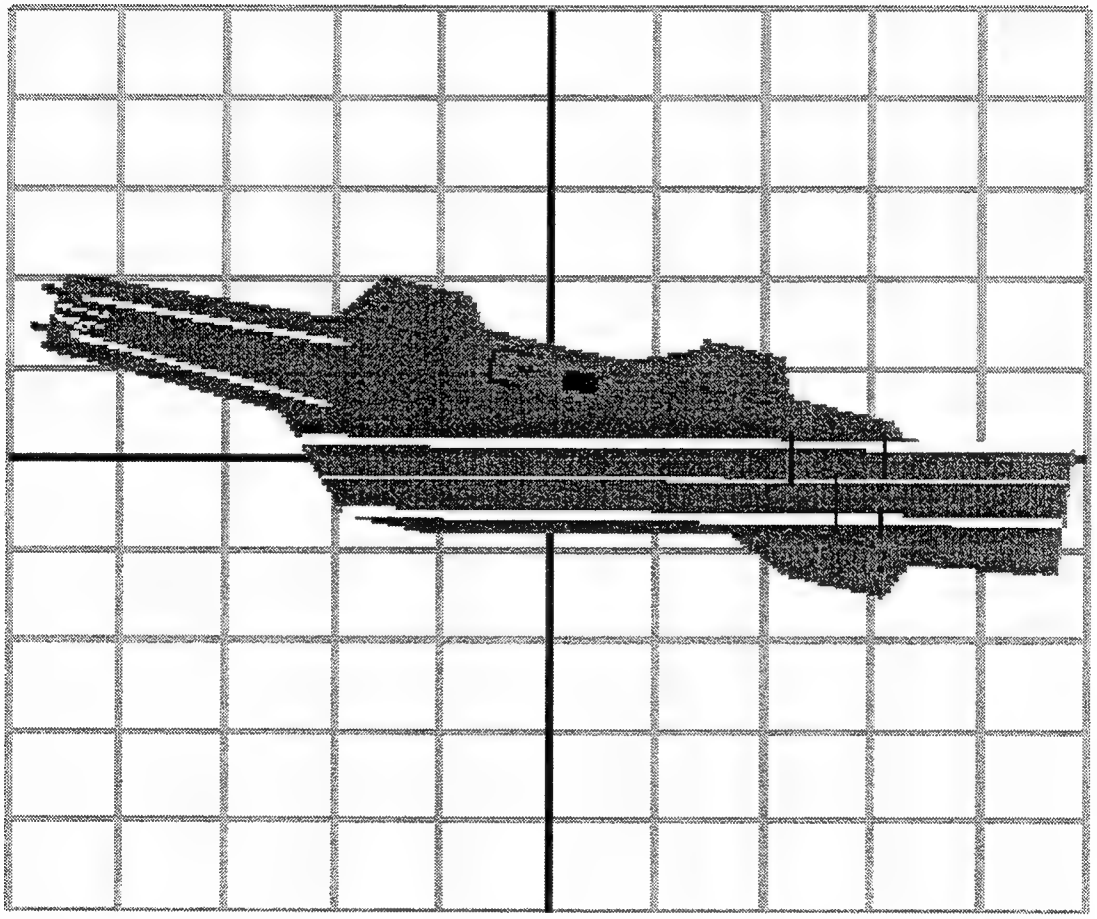


Figure A.2: USS Midway, CV-41

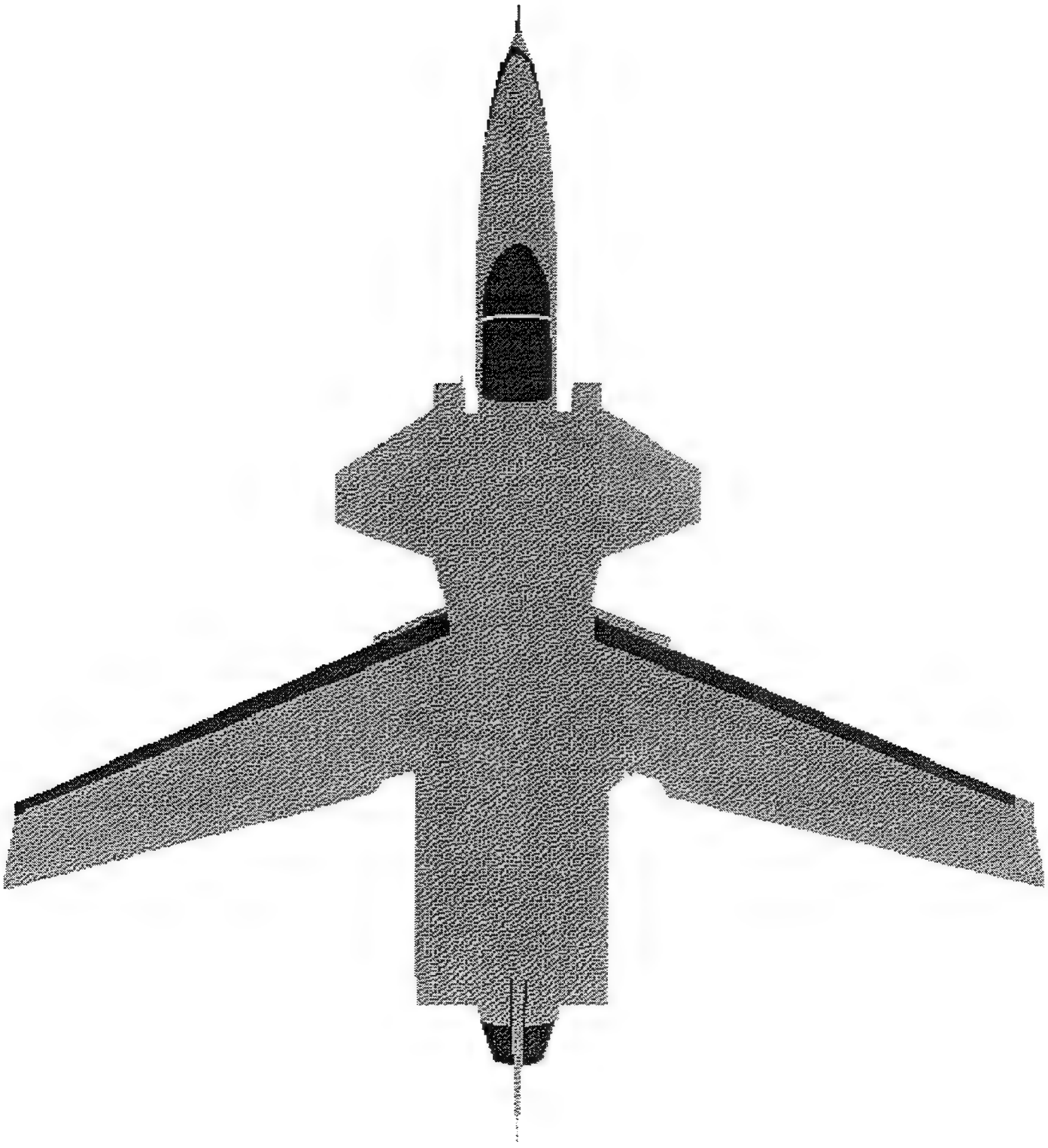


Figure A.3: Generic Jet Aircraft

Figure A.4: Aircraft and Carrier Initial Positions

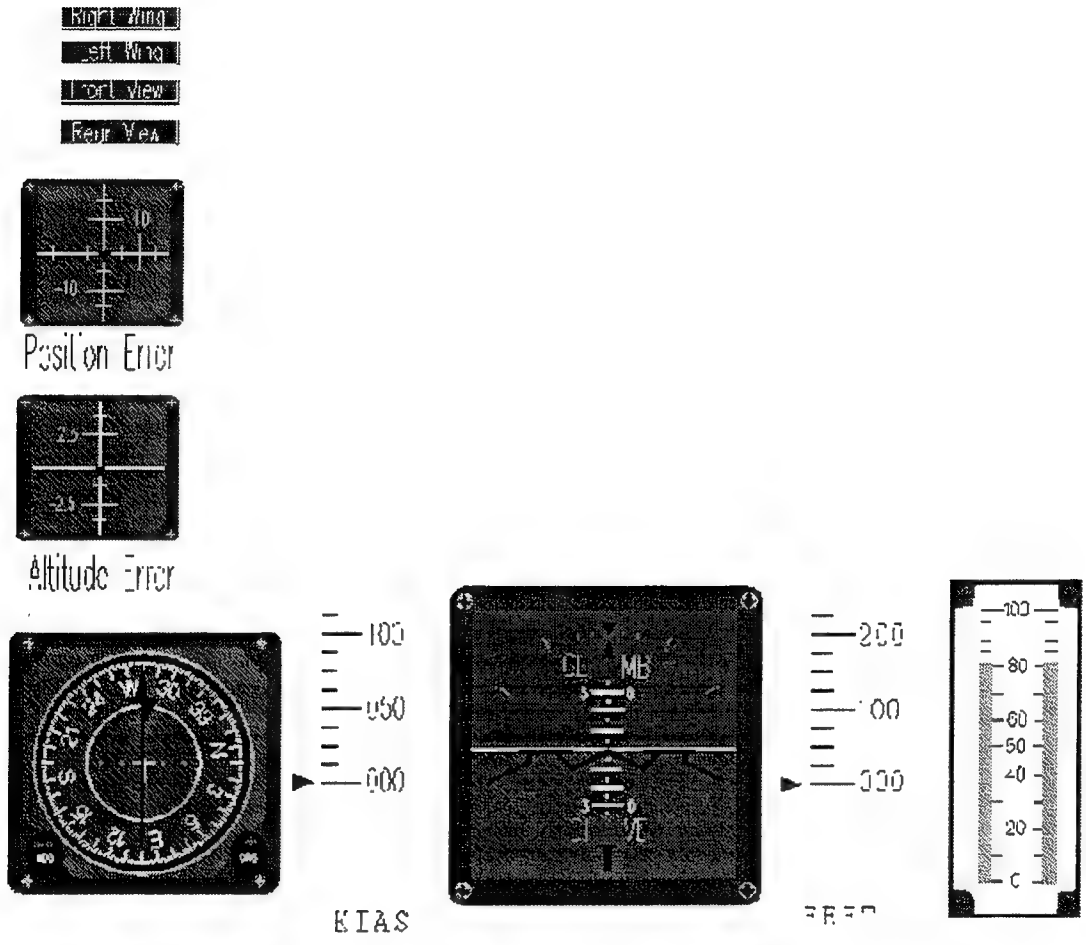


Figure A.5: Generic Cockpit

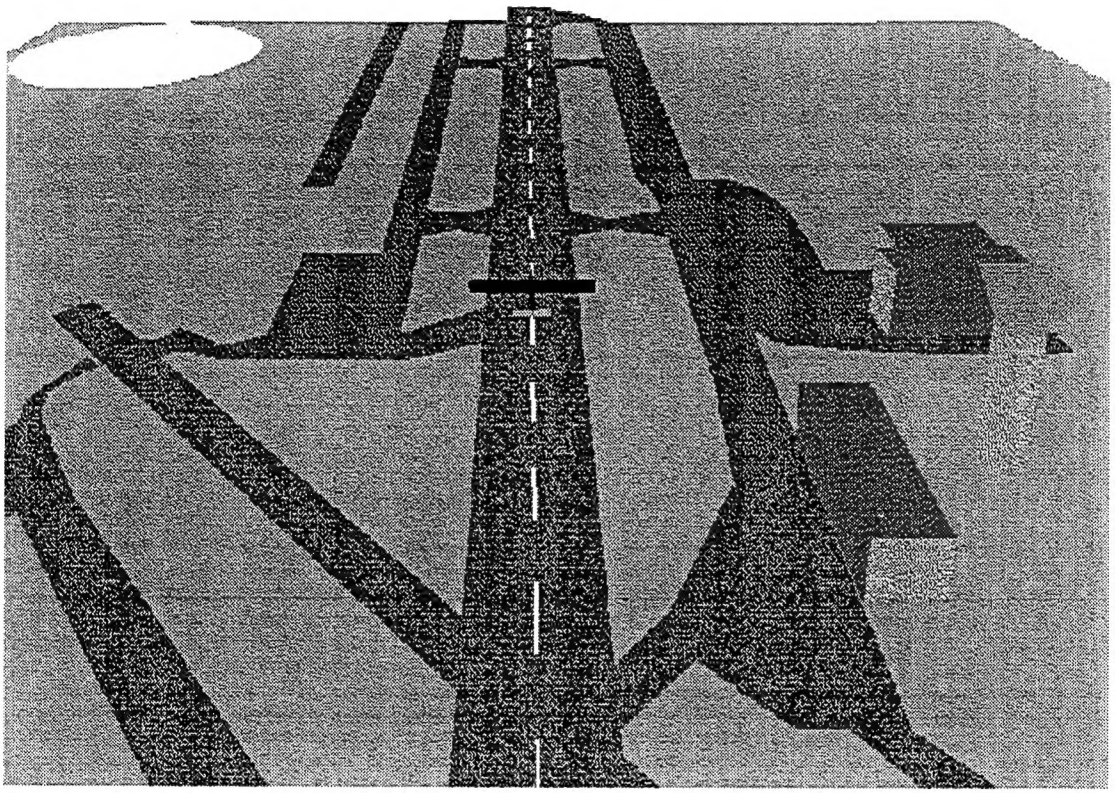


Figure A.6: Trajectory Controller : Aircraft Initial Position on Airfield

LIST OF REFERENCES

Coryphaeus Software, Inc., "Designer's Workbench *Users* Manual" June 1993.

Coryphaeus Software, Inc., "Designer's Workbench *Reference* Manual" June 1993.

Coryphaeus Software, Inc., "Designer's Workbench *Link Editor/Run Time Module User's* Manual June 1993.

Hallberg, E. N. , *Design of a GPS Aided Guidance, Navigation, and Control System for Trajectory Control of an Air Vehicle*, MSAE Thesis, Dept. of Aeronautics and Astronautics, Naval Postgraduate School, Monterey, CA, March 1994.

Lagier, M. , *An Application of Virtual Prototyping to the Flight Test and Evaluation of an Unmanned Air Vehicle*, MSAE Thesis, Dept. of Aeronautics and Astronautics, Naval Postgraduate School, Monterey, CA, March 1994.

Niewwoehner, R. J. , *Plant/Controller Optimization by Convex Methods*, PhD Thesis, Dept. of Aeronautics and Astronautics, Naval Postgraduate School, Monterey, CA, March 1994.

INITIAL DISTRIBUTION LIST

		No. of Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Dr. Isaac I. Kaminer Department of Aeronautics and Astronautics, Code AA/KA Naval Postgraduate School Monterey, CA 93943-5000	5
4.	Dr Richard M. Howard Department of Aeronautics and Astronautics, Code AA/HO Naval Postgraduate School Monterey, CA 93943-5000	1
5.	Chairman Department of Aeronautics and Astronautics Naval Postgraduate School Monterey, CA 93943-5000	2

No. of Copies

7. LT Arnold P Selnick — 1
   USS America
   OW Division
   FPO AE 09531-2790

8. Mr. L. L.Selnick — 1
   35 Topcrest Lane
   Ridgefield CT, 06877